

17th AIAA/CEAS Aeroacoustics Conference (32nd AIAA Aeroacoustics Conference), 6 – 8 June 2011, Portland, Oregon.

Hermite Methods for Aeroacoustics: Recent Progress

Daniel Appelö ^{*} Matthew Inkman[†] Thomas Hagstrom [‡] Tim Colonius [§]

We present recent developments on Hermite methods for aeroacoustic simulations including time-stepping methods, hybridization with discontinuous Galerkin methods for handling of boundary conditions and adaptive implementations. By scaling studies reported below we show that the features unique to Hermite methods have promise to enable efficient exploitation of modern petascale architectures. We also present preliminary computations of turbulent jet noise obtained with the current implementation of our compressible Navier-Stokes solver.

I. Introduction

Enhanced understanding of turbulence mixing noise has high scientific and technological value and will be realized by a combination of experimental, computational and theoretical studies. As computational resources grow more abundant Direct Numerical Simulation (DNS) and Large Eddy Simulation (LES) are candidates for direct prediction of the sound generated by large-scale turbulence. For this to be realized it is crucial to develop computational approaches that are amenable to implementation on massively parallel computer architectures. Also, as the acoustic radiation we are interested in is several orders of magnitude smaller than energetic turbulent structures, high order accurate methods are favored³ since they provide minimal dispersion and dissipation to both turbulent flow structures and their weak radiated sound.

We focus on a class of methods of arbitrary order of accuracy known as Hermite methods^{1,2} which, we believe, have many features that make them suitable for parallel architectures. Although the basic theoretical underpinnings of Hermite methods applied to hyperbolic systems have been known for some time,¹ there are many practical and theoretical aspects still to be investigated. Here we present some new results on time-stepping, how to handle boundary conditions by hybridization with discontinuous Galerkin methods, characteristic boundary conditions, adaptive implementations and efficient computation of forcing terms.

Our ultimate goal is to be able to perform direct numerical simulations of jets at significantly higher Reynolds numbers than the present state of the art DNS by Freund.¹⁴ To get there, we use a two pronged approach. On the one hand, we are developing and applying more sophisticated algorithms to one and two dimensional model problems. Simultaneously, on the other hand, we are developing a massively parallel 3D compressible Navier-Stokes solver that leverages the sophisticated algorithms from the first step as they become mature. Currently we are able to compute jet flows similar to Freund's Re 3600 jet on a moderately large number of cores. Below, we present preliminary results of a simulation of a Mach 0.90, Re 3600 jet that will be used to verify our code. At this preliminary stage (approximately 1.5 residence times) the solution is displaying similar ranges of scales and flow features as Freund's. More detailed comparisons are forthcoming as the flow becomes statistically stationary. We present strong scaling results indicating that the algorithm scales well for this moderate size problem and number of cores.

II. Description of the Method

This section describes the Hermite method with Taylor series time-stepping¹ and with Runge-Kutta time-stepping.² It also considers how to hybridize the method with nodal discontinuous Galerkin to handle boundary conditions, and presents results from a model example illustrating the good computation to

^{*}Postdoctoral Scholar, Department of Mechanical Engineering, California Institute of Technology.

[†]Graduate Student, Department of Mechanical Engineering, California Institute of Technology.

[‡]Professor, Department of Mathematics, Southern Methodist University, AIAA Senior Member.

[§]Professor, Department of Mechanical Engineering, California Institute of Technology, AIAA Senior Member.

communication ratio, the resolving power and the efficiency of the method. Adaptive implementations and a fast and accurate method for computation of source terms are also discussed.

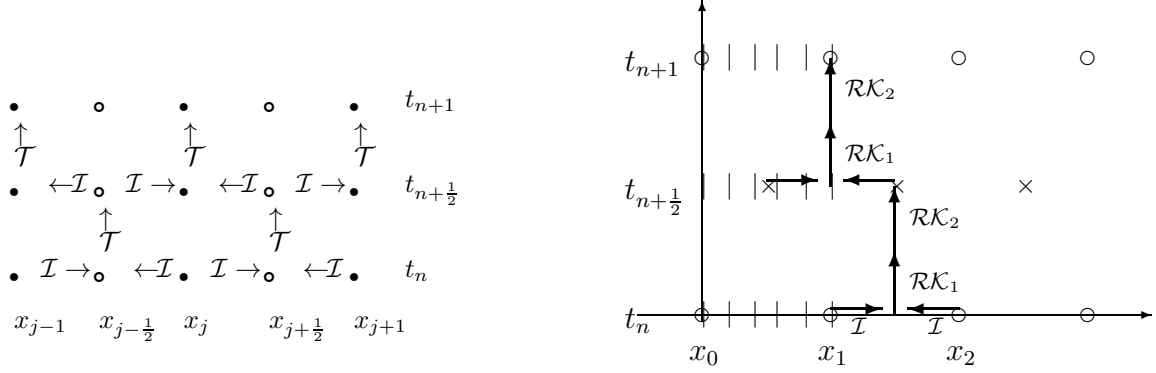


Figure 1. Left: Schematic description of the numerical process for a full time-step. Solid circles represent the base mesh and open circles represent the dual mesh. \mathcal{I} is the Hermite interpolation operator and \mathcal{T} is the time evolution operator. Right: Schematic picture of the hybrid Hermite-DG method.

A. Hermite Taylor Algorithm

For simplicity, consider the approximation of the advection equation:

$$u_t = u_x, \quad (1)$$

in one dimension. Let $x_j = j \Delta x$ be grid points on a uniform grid. The degrees of freedom in a (1D) Hermite method are coefficients c_{l0} approximating the scaled spatial derivatives

$$c_{l0} \approx \frac{1}{l!} \frac{\partial^l u(x_j, t)}{\partial x^l}, \quad l = 0, \dots, m, \quad (2)$$

at each grid point (see Figure 1). At the start of a time-step, $t = t_n$, the $2m + 2$ coefficients in two adjacent grid points are used to form a *local* power series, in the form of an interpolating Hermite polynomial, centered around the midpoint $x_{j+1/2}$ of element $x \in (x_j, x_{j+1})$:

$$p_{j+1/2}(x, t_n) = \sum_{l=0}^{2m+1} c_{l0} (x - x_{j+1/2})^l. \quad (3)$$

Expansion of the coefficients c_{l0} in time by a Taylor series around t_n yields

$$p_{j+1/2}^n(x, t) = \sum_{l=0}^{2m+1} \sum_{s=0}^{2m+1-l} c_{ls} (x - x_{j+1/2})^l (t - t_n)^s. \quad (4)$$

The polynomial (4) could be evaluated, and the approximate solution advanced, if the coefficients c_{ls} were known. We find these by taking spatial and temporal derivatives of the evolution equation (1):

$$\frac{\partial}{\partial t} \frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial t^2} = \frac{\partial^2 u}{\partial t \partial x} = \frac{\partial^2 u}{\partial x^2}, \quad \Rightarrow \quad \frac{\partial^s u}{\partial t^s} = \frac{\partial^{s-1} u}{\partial t^{s-1}} \frac{\partial u}{\partial x}. \quad (5)$$

Inserting $p_{j+1/2}(x, t) \approx u(x, t)$ into (5) and evaluating the resulting equation at $(x_{j+1/2}, t_n)$ produces:

$$c_{ls} = \frac{l+1}{s} c_{l+1, s-1}, \quad l = 0, \dots, 2m+1-s, \quad s = 1, \dots, 2m+1. \quad (6)$$

By (6), equation (4) can be evaluated at time $t = t_{n+1/2}$ to find c_{ls} on the dual grid. The process is repeated to complete the evolution of the scaled derivatives to time $t = t_{n+1}$.

B. Runge-Kutta Time Stepping

For linear problems it is advantageous to evaluate all the terms in the series (4) resulting in a method with truncation errors $\sim (\Delta t^{2m+1} + \Delta x^{2m+1})$. For nonlinear and variable coefficient problems the recursion relation for the coefficients (5) becomes more involved and expensive to evaluate.² In particular for large m and s , higher order accurate Taylor methods do not justify the additional computational expense. For our 3D-CNS solver we instead use explicit Runge-Kutta methods that only require evaluation of the least expensive term, $s = 1$, in the non-linear version of (6).

The resulting evolution equations, $\partial_t c_{ls} = \mathcal{F}$, are ordinary differential equations and local on each element. Thus, communication with neighboring grid points is only necessary when dictated by the domain of dependance. This locality is translated into a greater computation to communication ratio and higher accuracy by performing multiple smaller local Runge-Kutta sub-steps $\delta t = \Delta t / n_{\text{sub}}$, before communicating. This is in sharp contrast with most other methods where communication is required between every Runge-Kutta stage. The high computation to communication ratio is central to the good scalability of Hermite methods. Below, in Section D, we illustrate the effect of local sub-stepping on communication for a model example.

We note that the locality also has ramifications for memory usage. If, for example, an r -stage Runge-Kutta method is used, only the local r evaluations of \mathcal{F} have to be stored. This is to be compared to the r global arrays required to store corresponding (global) right hand side evaluations for a finite difference method. The difference for a 3D problem can be significant.

C. Boundary Conditions

Hybrid Hermite-Runge-Kutta Discontinuous Galerkin Method

Hermite methods use derivative data as degrees of freedom, and special care is needed to treat the boundary conditions.¹ The approach in Ref. 1 and its generalization to complex geometries is somewhat involved. Here we describe an alternative way to impose boundary conditions by coupling a nodal discontinuous Galerkin (DG) method,⁴ used to evolve the solution next to the boundary, to the Hermite-Runge-Kutta method.

For simplicity, consider a one-dimensional uniform grid $x_0, x_1 = x_0 + \Delta x, \dots$, with a boundary at $x = x_0$, see Figure 1. A Legendre-Gauss-Lobatto (LGL) grid is introduced in $x_0 \leq x \leq x_1$. On this grid (element) a nodal discontinuous Galerkin method with an upwind flux for the element boundary conditions is used. To the left the physical boundary conditions are used to compute the numerical flux and to the right Hermite data is used for the flux. To be precise, a full time-step starts by advancing the Hermite solution on the dual grid $x_{\frac{1}{2}}, x_{\frac{3}{2}}, \dots$ using some number of local Runge-Kutta substeps. At the start and at the end of each substep we extrapolate u and u_t to x_1 to find a fourth order accurate Hermite time-interpolant of $u(x_1, t)$. The interpolant is used as boundary data for the time evolution of the DG data. This evolution is also performed by a Runge-Kutta method with a (larger) number of substeps. Once the Hermite and the DG data have been advanced to time $t_{n+\frac{1}{2}}$ the Hermite data at x_1 is obtained by differentiating the DG data. The second half step is similar to the first, the exception being that no extrapolation of u, u_t is needed.

To demonstrate the approach described above we solve

$$u_t = u_x, \quad v_t = -v_x, \quad t \geq 0, \quad -1 \leq x \leq 1, \quad (7)$$

with boundary conditions and initial data:

$$\begin{aligned} u(-1, t) &= v(-1, t), \quad u(1, t) = v(1, t), \\ u(x, 0) &= v(x, 0) = \sin(\omega_0 \pi x), \end{aligned}$$

until time 40, when we measure the maximum error. The computations are performed on a grid with $\Delta x = 0.1$ and with $\omega_0 = 12.5$. In Table 1 we present results for a 7th order accurate Hermite method ($m = 3$) combined with an 8th order accurate DG method. Both methods are advanced using the classic fourth order accurate Runge-Kutta methods with different numbers of substeps. Fixing the number of substeps in the Hermite and DG methods we find the minimal number of global time-steps of size Δt such that the solution is stable. The errors at this time-step combinations are reported in Table 1. As can be seen $\text{CFL} \equiv \Delta t / \Delta x$ is increased by taking more substeps. As expected the DG method requires smaller time-steps than the Hermite method. Finally, we note that the hybrid-method generalizes to multi-D. As an example to test the accuracy and stability we solve the wave equation $u_t + v_x - u_y = 0$, $v_t + u_x + v_y = 0$ with $u(x, y, 0) = v(x, y, 0) = \sin x \sin y$, $(x, y) \in [-\pi, \pi]$, $t > 0$. We measure the maximum error at time $t = 200\pi$ for $m = 2, 3, 4$, which is displayed in Table 2.

Table 1. Maximum error for a one dimensional problem with $m = 3, n_{DG} = 7$.

$n_{\text{sub.H.}}$	$n_{\text{sub.DG}}$	$\Delta t/\Delta x$	$\Delta t/\Delta x/n_{\text{sub.}}$	Error
1	2	0.40	0.40	1.0 (0)
1	3	0.63	0.63	4.5 (-1)
1	4	0.69	0.69	1.8(-2)
2	2	0.40	0.20	1.6 (-1)
2	6	0.84	0.42	1.4 (-2)
3	3	0.60	0.20	1.1(0)
3	6	0.93	0.31	4.7(-3)

Table 2. Convergence for the two dimensional hybrid method.

m, n_{DG}	Δx	$\Delta t/\Delta x$	$n_{\text{sub.H.,DG}}$	Error	Rate
2, 4	$\pi/10$	0.87	2, 15	1.21(-2)	-
2, 4	$\pi/20$	0.87	2, 15	2.93(-4)	5.4
2, 4	$\pi/40$	0.87	2, 15	3.09(-6)	6.6
3, 7	$\pi/20$	0.71	2, 20	2.23(-3)	-
3, 7	$\pi/40$	0.71	2, 20	1.46(-5)	7.3
3, 7	$\pi/80$	0.71	2, 20	2.79(-7)	5.7
4, 8	$\pi/6$	0.6	2, 20	8.45(-3)	-
4, 8	$\pi/12$	0.6	2, 20	3.23(-4)	4.7
4, 8	$\pi/24$	0.6	2, 20	1.15(-5)	4.8

D. Efficiency and Resolving Power

We start by illustrating how local sub-stepping can be used to delay the need for communication until it is required by the domain of dependence boundary. To do this we solve (1) on $x \in [-0.5, 0.5]$ with periodic boundary conditions and initial data $u(x, 0) = \sin 2\pi x$ until time $t = 1$.

In Figure 2 we have plotted maximal stable $\Delta t/\Delta x$ for different number of sub-steps $n_{\text{sub.}} = 1, 4, 16$ and for methods of different order, $m = 1, \dots, 9$. As can be seen, the CFL number $\Delta t/\Delta x$ approaches the optimal value of 1 with larger $n_{\text{sub.}}$ for all m , substantiating our claim that we can achieve a high computation to communication ratio with this method. Of course, it is not necessary to require the substeps have equal size. They can also be chosen adaptively, e.g. using some commercial or open source ODE software.

When the problem is not purely hyperbolic, the maximal time-step is no longer just a function of the domain of dependence. In particular, if the evolutionary equation contains a parabolic term, say $u_t + Uu_x = \nu u_{xx}$, the bound on the time-step will be $\Delta t \leq \Delta t_{\text{com.}} \leq C \min(\Delta x/U, \frac{\Delta x^2}{\nu m})$. Thus for large viscosity or small Δx the parabolic part will set the maximum Δt . In that limit sub-stepping will not necessarily delay the need for communication, see Figure 2. It will, however, still make the solution more accurate. As can be seen in Figure 2, the methods of different order behave somewhat differently in the presence of a viscous term. This impacts the choice of m when applying it to low Re flows and to large eddy simulations.

For the simulations we target, we anticipate resolving the flow such that the grid Reynolds number $\text{Re}_{\text{grid}} = U\Delta x/\nu$ is $\mathcal{O}(10)$. Rearranging the terms, $\Delta t \leq \Delta t_{\text{com.}} \leq C \frac{\Delta x}{U} \min(1, \frac{\text{Re}_{\text{grid}}}{m})$, we see that the time-step will be limited by the hyperbolic constraint unless m is very large.

Next, to compare the efficiency and resolving power of the DG-Hermite hybrid method to two more traditional finite difference methods, we again solve (7). For this example we set $\omega_0 = 7.5$ and solve to time $t = 16$. For the DG-Hermite scheme we use Taylor time-stepping in the Hermite cells and classic Runge-Kutta for the DG elements. We present results for various orders in space but make sure to match the order of the DG and the Hermite methods.

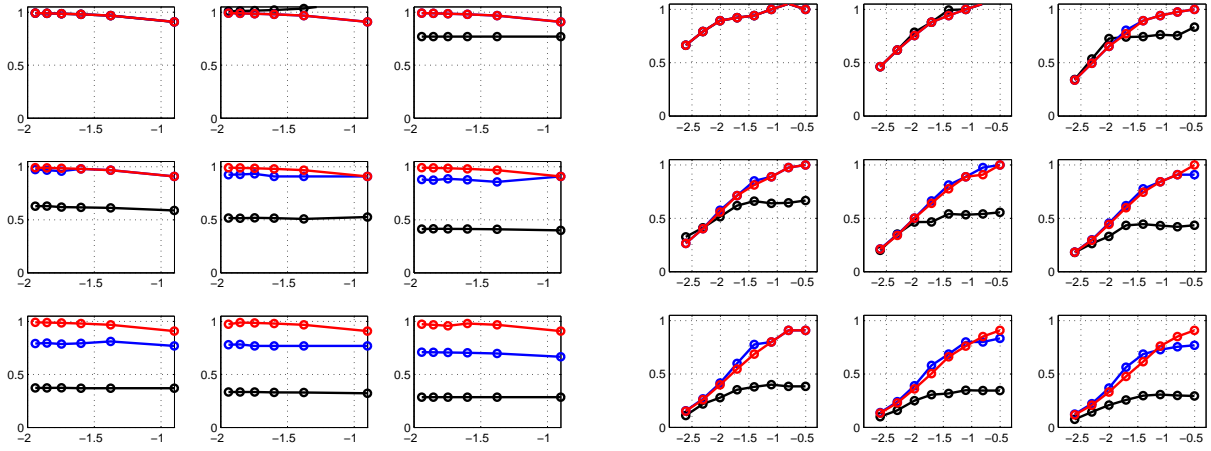


Figure 2. The plots display the maximal stable $\Delta t/\Delta x$ (on the vertical axis) for $n_{\text{sub}} = 1$ (black), 4 (blue), 16 (red) for different $\log_{10} \Delta x$ (on the horizontal axis). The left subfigure is for $u_t + u_x = 0$ and the right is for $u_t + u_x = \frac{1}{2000} u_{xx}$. The 9 subplots in each subfigure are for, from upper left to lower right $m = 1, \dots, 9$.

The two finite difference methods we compare against are the classic fourth order accurate Pade scheme⁵ with fourth order closures at the boundary and a summation-by-parts⁶ (SBP) scheme using the 4-8 diagonal norm operators at the boundaries (see appendix C.4 in ref. 7). Both the schemes use the classic fourth order accurate Runge-Kutta method in time.

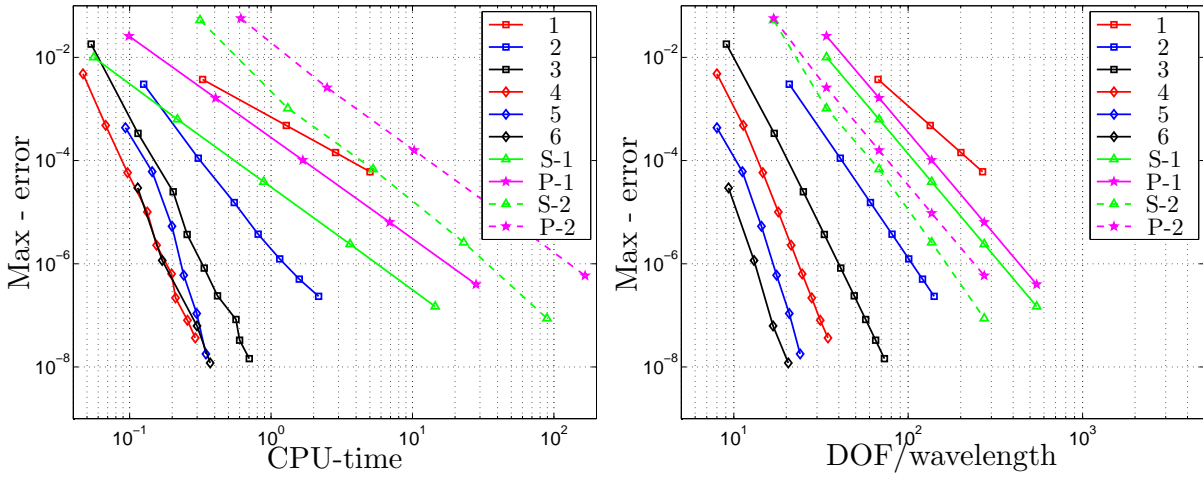


Figure 3. In the legend the numbers indicate $m = 1, \dots, 6$. S & P denote summation by parts and Pade. The trailing numbers for S & P indicate a time-step close to the stability limit (-1) or very small time-step (-2).

For the finite difference methods we consider two regimes. In the first (denoted S-1 & P-1 in Figure 3) we try to compare efficiency and thus keep the time-step close to the stability limit. In the second (denoted S-2 & P-2) we study the resolving power and reduce the time-step by a factor of 20 so that the spatial error is dominant. For the DG-Hermite-Taylor method we always keep the time-step close to the stability limit.

To the left in Figure 3 the maximum error is plotted against the time it takes (start to end) to complete the computation. We observe that for errors smaller than 0.1% the DG-Hermite methods of order greater than 3 are more efficient than the finite difference methods. Obviously the efficiency of the methods will depend on the computer (in this case a MacBook 2GHz Intel Core 2 Duo), yet the overall trend should be the same. For this hardware it appears that the 9th order method is most efficient over a wide range of accuracies. It is important to note that the comparison is for a linear equation. As stated above, the computational complexity for a non-linear problem increases with m thus the optimal choice of m could be different and the superior efficiency relative the finite difference methods might be less pronounced.

To the right in Figure 3 the maximum error is plotted against the number of degrees of freedom per wavelength. For the finite difference method the degrees of freedom (DOF) equals the number of grid points while for the DG-Hermite methods the DOF are the number of grid points multiplied by $(m + 1)$. From the figure we conclude that the resolving power of the DG-Hermite method is good compared to the finite difference methods. This is expected for the higher order methods $m > 2$, but, more surprisingly, is also true for the $m = 2$ fifth order accurate method.

Outflow and Characteristic Boundary Conditions

To damp outgoing acoustic disturbances in nonlinear flow problems we usually apply some form of the super-grid-scale damping layer.^{2,8,9} If such layers are applied around a quiescent or uniform flow state it is convenient to terminate the layer with periodic conditions. However, in the case of a jet the downstream and upstream flow conditions are very different and periodic boundary conditions can cause significant leakage and drift. In such cases it is more suitable to truncate the damping layer with a characteristics based boundary condition.

Of course, as alluded to above, the difficulty with imposing boundary conditions in a Hermite method is primarily due to the fact that additional equations governing the spatial derivatives on the boundary have to be derived. This is true for an outflow boundary as well but there is some additional freedom. We take the approach that any reasonable outflow boundary condition should be invariant to translation. Therefore, for inviscid systems, the standard characteristic boundary conditions, $R_{\text{in}} = 0$, generalize to

$$\frac{\partial^l R_{\text{in}}}{\partial n^l} = 0, \quad l = 0, \dots, m. \quad (8)$$

Here ∂_n is the derivative in the direction normal to the boundary and R_{in} are the incoming characteristic variables.

E. Adaptive Implementations

hp-adaptivity

Many of the unsteady compressible flow problems we plan to study with our code will exhibit small regions requiring enhanced resolution (for example shear layers near a nozzle) or large regions admitting coarsened resolution (for example regions of laminar flow bounding a turbulent jet). To efficiently handle these situations we are implementing an *hp*-refinement capability.

The unique features of Hermite methods make order refinement particularly straightforward to implement. As the global time-step for hyperbolic problems can be chosen independently of the local polynomial degree, no special treatment is required to evolve polynomials of differing degree on different cells. The only restriction is that the interpolation between the staggered grids must maintain stability. We show in Ref. 10 how this may be accomplished by requiring that all unidirectional interpolations use polynomial data of the same degree at each node. For example, in one space dimension, if the polynomial degree is m_k at x_k and m_{k+1} at x_{k+1} we set $\bar{m}_{k+1/2} = \min\{m_k, m_{k+1}\}$ and construct the degree $2\bar{m}_{k+1/2} + 1$ Hermite interpolant of the function values and derivatives through order $\bar{m}_{k+1/2}$ at each node. We can easily prove that this interpolation step is stabilizing, and experiments in Ref. 10 demonstrate its effectiveness in one and two space dimensions.

Implementation of grid refinement, on the other hand, is accomplished using uniform space-time refinements of the computational cells. A brief description of the process in the simplest case of one refinement level is as follows:

- i. Assuming both levels have been advanced to time t on the base grid, advance each level on interior nodes of the dual grid a half step.
- ii. Having completed these half steps, we have dual grid data at time $t + \Delta t/2$ on the coarse grid and $t + \Delta t/4$ on the fine grid. Now carry out another half step on the fine grid. For nodes belonging only to refined cells, this is straightforward. For nodes on a refinement border, on the other hand, we update the data by interpolation rather than by evolution.

- iii. Carry out one half step on the coarse grid and two half steps on the refined grid. Nodes on the refinement border belonging to both time levels are evolved on the coarse grid half step, while hanging nodes are updated by interpolation. This step returns us to (i.).

This procedure is easily extended to multiple nested levels; that is, refinements with the property that bordering cells can only be at most one refinement level apart. We also note that each partial time-step we have described can be accomplished using multiple substeps of some Runge-Kutta method as described above.

To illustrate the stability and accuracy of this approach we solve

$$\begin{aligned} u_t + (1 + M \cos \theta)u_x + v_y &= 0, \\ v_t - (1 - M \sin \theta)v_x + u_y &= 0, \\ (x, y) &\in [-1, 1] \times [-1, 1], \quad 0 \leq t \leq 100, \end{aligned} \quad (9)$$

for the 2-periodic solution

$$u = \cos(\omega\pi t) \cdot \cos(k_1\pi(x - M \cos \theta t)) \cdot \sin(k_2\pi(y - M \sin \theta t)), \quad (10)$$

$$\begin{aligned} v &= -\frac{\omega}{k_2} \sin(\omega t) \cdot \cos(k_1\pi(x - M \cos \theta t)) \cdot \cos(k_2\pi(y - M \sin \theta t)) \\ &\quad - \frac{k_1}{k_2} \cos(\omega t) \cdot \sin(k_1\pi(x - M \cos \theta t)) \cdot \cos(k_2\pi(y - M \sin \theta t)). \end{aligned} \quad (11)$$

Here $M = .5$, $\theta = \frac{\pi}{3}$, $k_1 = 5$, $k_2 = 6$ and $\omega = \sqrt{61}$, so that we are solving over 390 periods. We use three nested grid levels; the finest grid, with cells and time-steps one fourth of the coarse grid, is located on $[-.25, .25] \times [-.25, .25]$. The first refinement level is located on $([-.5, .5] \times [-.5, .5]) - ([-.25, .25] \times [-.25, .25])$ and the coarse grid on $([-1, 1] \times [-1, 1]) - ([-.5, .5] \times [-.5, .5])$. The results for $m = 3 - 5$ and order $2m + 2$ Taylor time-stepping are shown in Table 3. We clearly observe long time stability and convergence at the design order.

m	Δx_{coarse}	Δt_{coarse}	CFL	error	rate
3	$\frac{1}{10}$	$\frac{1}{18}$	0.833	2.59×10^{-2}	
3	$\frac{1}{20}$	$\frac{1}{36}$	0.833	2.08×10^{-4}	7.0
3	$\frac{1}{30}$	$\frac{1}{54}$	0.833	1.25×10^{-5}	6.9
4	$\frac{1}{8}$	$\frac{1}{14}$	0.857	1.36×10^{-3}	
4	$\frac{1}{12}$	$\frac{1}{21}$	0.857	4.04×10^{-5}	8.7
4	$\frac{1}{16}$	$\frac{1}{28}$	0.857	3.20×10^{-6}	8.8
5	$\frac{1}{6}$	$\frac{1}{14}$	0.643	4.06×10^{-4}	
5	$\frac{1}{10}$	$\frac{1}{22}$	0.682	1.59×10^{-6}	10.9
5	$\frac{1}{12}$	$\frac{1}{28}$	0.643	2.29×10^{-7}	10.6

Table 3. Convergence at $t = 100$ for problem (9) with 3 refinement levels.

Grid Stretching

The hp-refinement approach described above is very efficient but has not been implemented in our 3D-CNS solver yet. As an intermediate step we have instead used stretched grids to increase the resolution in regions where the flow is changing rapidly. We use the transport equation to illustrate. Let the grid stretching is governed by the mapping $x = x(s)$, from the uniform grid in s to the stretched grid in x . Application of the chain rule yields a variable coefficient problem

$$\frac{\partial u(x(s))}{\partial t} = \frac{\partial u(x(s))}{\partial x} = \underbrace{\left(\frac{\partial x(s)}{\partial s} \right)^{-1}}_{a(s)} \frac{\partial u}{\partial s}. \quad (12)$$

Thus, for the l th derivative of (12) we get

$$\frac{\partial^l}{\partial s^l} \frac{\partial u(x(s))}{\partial t} = \frac{\partial^{l+1} u(x(s))}{\partial t \partial s^l} = \frac{\partial^l}{\partial s^l} \left(a(s) \frac{\partial u}{\partial s} \right). \quad (13)$$

The recursion relation for (13) can be derived using Leibniz's rule as described in ref. 1. As the equations are approximated in the equidistantly discretized variable s , while the initial data is given in the physical coordinate x , we must employ Faà di Bruno's formula to rescale the initial data accordingly. For example, the first three terms become:

$$\frac{\partial u(x(s))}{\partial s} = \frac{\partial x(s)}{\partial s} \frac{\partial u}{\partial x}, \quad (14)$$

$$\frac{\partial^2 u(x(s))}{\partial s^2} = \left(\frac{\partial x(s)}{\partial s} \right)^2 \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 x(s)}{\partial s^2} \frac{\partial u}{\partial x}, \quad (15)$$

$$\frac{\partial^3 u(x(s))}{\partial s^3} = \left(\frac{\partial x(s)}{\partial s} \right)^3 \frac{\partial^3 u}{\partial x^3} + 3 \frac{\partial x(s)}{\partial s} \frac{\partial^2 x(s)}{\partial s^2} \frac{\partial^2 u}{\partial x^2} + \frac{\partial^3 x(s)}{\partial s^3} \frac{\partial u}{\partial x}. \quad (16)$$

F. Computation and Differentiation of Source Terms

In order to facilitate the use of a variety of functions for input forcing and buffer region damping, we have implemented an adaptive bivariate Chebyshev approximation routine in our code. The purpose of this algorithm is to allow for easy computation of the $2m + 1$ orders of partial derivative data necessary to apply arbitrary forcing terms to equations being solved with the Hermite method. Our work adapts the univariate Chebyshev approximation algorithm of the Chebfun Team¹¹ to model bivariate functions, following the method of Sommariva, Vianello, and Zanollo.¹² The Chebfun algorithm was translated into Fortran and applied with minor adaptations to the bivariate problem.

The univariate Chebyshev series expansion of a function takes the form $f(x) = \sum_{i=0}^{\infty} c_i T_i(x)$, $x \in [-1, 1]$. Similarly, a bivariate function $f(x, y)$ can be expanded in y such that $f(x, y) = \sum_{i=0}^{\infty} c_i(x) T_i(y)$ on the square $[-1, 1]^2$. The adaptive algorithm produces an approximation of a bivariate function by expanding $c_i(x)$ as a Chebyshev series in x , producing a result that can then be mapped to a specified rectangular domain. At the first step ($k = 0$), a set of Chebyshev-Lobatto nodes, $\zeta_i = \cos(i\pi/m_k)$, $i = 0, \dots, m_k$ are taken in x . A vertical cut is then made at each node, along which the univariate algorithm is applied to obtain a truncated Chebyshev expansion in y , $\sum_{i=1}^{n_j} c_i(x) T_i(y)$ meeting a specified error tolerance along each cut $x = \zeta_j$. The approximation that employs the largest number of Chebyshev coefficients, n_{max} , is then determined, and the coefficient series of all $x = \zeta_j$ are padded with zeros so that all $n_j = n_{max}$. Then, for $c_i(x)$, $i = 1, \dots, n_{max}$, the univariate algorithm is again applied to determine coefficients for a polynomial expansion of $c_i(x)$ sampled at m_k Chebyshev-Lobatto points. This yields the Chebyshev coefficients c_{ij} , such that

$$f(x, y) \approx \sum_{i=1}^{n_{max}} \sum_{j=1}^{m_k} c_{ij} T_j(x) T_i(y).$$

The error of the bivariate approximation is then checked at specified points or the point of largest gradient in x and y . If the error exceeds the specified minimum, the algorithm is executed again ($k = k + 1$), doubling the number of Chebyshev-Lobatto nodes in x , $m_k = 2m_{k-1}$ and retaining the coefficient information c_i from nodes used in the previous iteration. Once a sufficiently accurate polynomial approximation has been obtained, series coefficients for $2m + 1$ orders of mixed partial derivatives are obtained using the recursion formula for derivatives of Chebyshev expansions. The values of the Chebyshev expansion of $f(x, y)$ and its mixed partial derivatives are then evaluated throughout the domain using barycentric Lagrange interpolation.¹³

III. Preliminary Simulations of Turbulent Jets

We conclude with some preliminary results from our 3D compressible flow solver. As a first large scale computation and for verification purposes we consider Freund's¹⁴ $\text{Re} = \rho_j U_j / \mu_j = 3600$ jet at Mach 0.9. In the plots below, lengths have been non-dimensionalized by jet diameter, $x = \frac{x^*}{D_j}$, pressure by density and

sound speed in the quiescent flow, $p = \frac{p^*}{\rho_\infty a_\infty^2}$, and vorticity by jet exit velocity and radius, $\omega = \frac{D_j}{2} \frac{\omega^*}{U_j}$, where * denotes a dimensional quantity.

The equations are solved in a computational domain consisting of a box $(x/D_j, y/D_j, z/D_j) \in [-1, 19] \times [-5.5, 5.5]^2$. Towards the end of the domain we add a zeroth order damping layer to absorb outgoing disturbances. The layer is terminated using characteristic boundary conditions as described above. The nozzle is modeled by the same momentum forcing as in ref. 14.

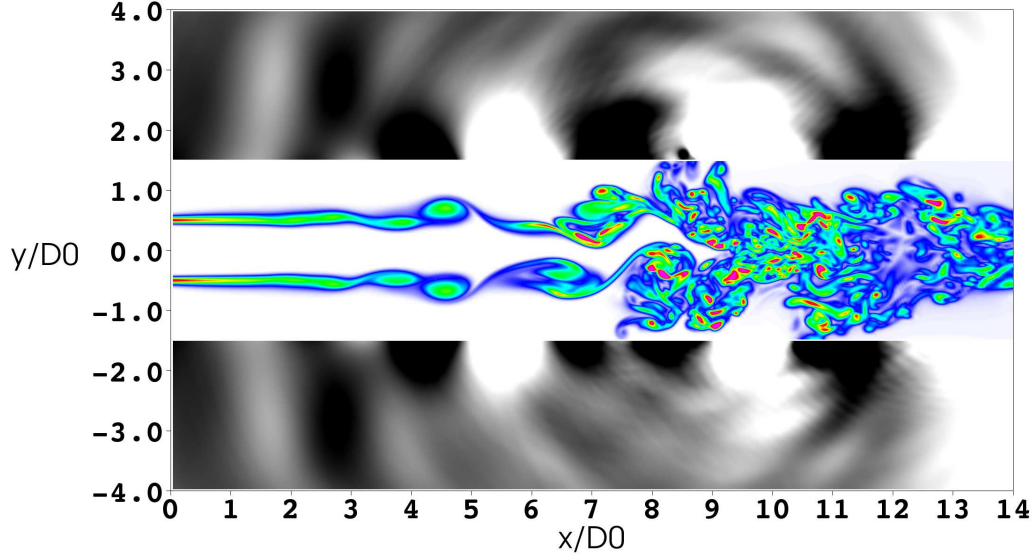


Figure 4. Snapshot of the vorticity magnitude and the pressure perturbation. The color scheme ranges from $\frac{D}{2} \frac{|\nabla \times u|}{U_{jet}} = 0, 10$ and $p - p_\infty = \pm 0.0025 p_\infty$.

To discretize the equations we use a seventh order accurate method in space and the classic fourth order accurate Runge-Kutta method in time. The computational domain is discretized on a stretched tensor grid consisting of $161 \times 126 \times 126$ points. The grid has a minimal grid spacing roughly four times larger than Freund, but as we carry 4^3 DOF per grid point the resolutions is expected to be comparable.

Figure 4 shows a snapshot of the vorticity and the pressure perturbation. Though the jet has not yet reached its statistical steady state at this point in its development it is still possible to observe the range of scales expected in a jet transitioning into turbulent flow. The pressure perturbation field is beginning to display traces of the convective multipole structure generally observed for a sound generating jet.

Figure 5 displays a snapshot of the contours of the vorticity. The contour levels are the same as in Figure 4. in ref. 14 enabling direct comparison. Again, as our jet is not fully developed the downstream spread is not as wide and the potential core is slightly longer than for Freund. But in both cases the evolution appears to be converging towards that of Freund.

Scaling

The simulations described were performed on Ranger at the Texas Advanced Computing Center (TACC) at University of Texas at Austin. The solver is parallelized by Cartesian domain decomposition where the communication steps are handled by standard MPI calls. To test the efficiency and strong scaling for the code we have performed a scaling study on 32-512 cores. The results are reported in Figure 6. The results are reasonably good and we expect that the results will be similar once we go to larger problems.

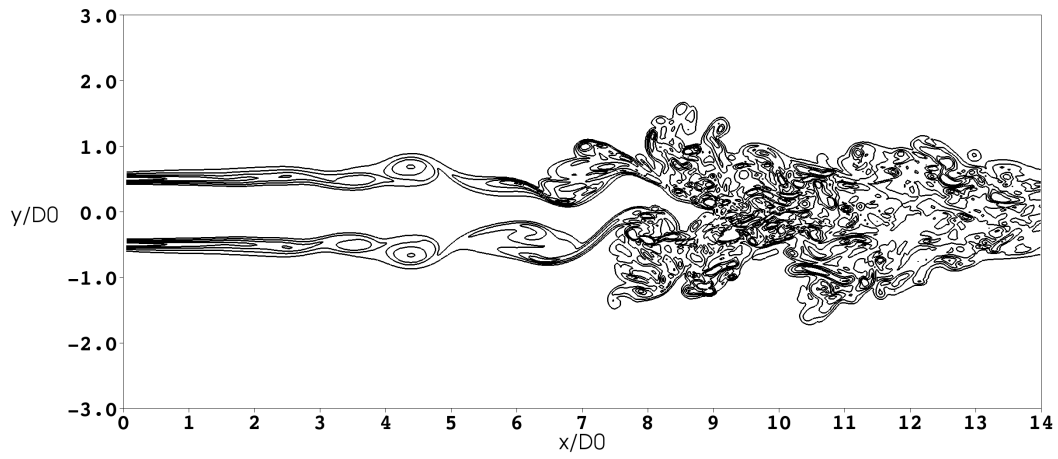


Figure 5. Snapshot of the vorticity magnitude at levels $\frac{D}{2} \frac{|\nabla \times u|}{U_{\text{jet}}} = 0.35, 1, 2, 3, 4$.

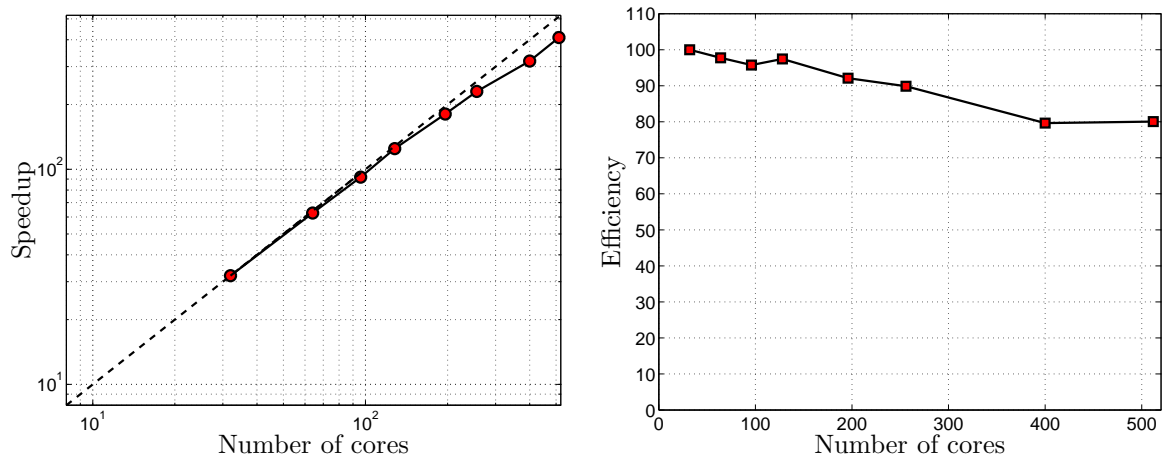


Figure 6. Strong speedup (left) and efficiency (right) for the Re 3600 jet computation.

IV. Conclusions

We have presented new developments on Hermite methods that once implemented in our 3D-CNS solver will permit the direct numerical simulation of higher Re jets. We have demonstrated that Hermite methods compare favorably to some classic finite difference methods both in terms of efficiency and resolving power. We have also shown that Hermite methods can be designed to have a very high computation-to-communication ratio, making them ideal for parallel implementation. We also presented preliminary simulation results of a turbulent jet. Our first DNS computation, a verification against Freund,¹⁴ is underway. Preliminary results are encouraging and more detailed results are forthcoming.

This material is based upon work supported, in part, by the National Science Foundation under Grants OCI-0905045 and OCI-0904773 and, in part, by ARO grant W911NF-09-1-0344. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation

References

- ¹Goodrich, J., Hagstrom, T., and and, J. L., “Hermite methods for hyperbolic initial-boundary value problems,” *Math. Comp.*, Vol. 75, No. 254, 2005, pp. 595–630.
- ²Hagstrom, T. and Appelö, D., “Experiments with Hermite Methods for Simulating Compressible Flows: Runge-Kutta Time-Stepping and Absorbing Layers,” *13th AIAA/CEAS Aeroacoustics Conference*, No. 2007-3505, 2007.
- ³Colonus, T. and Lele, S., “Computational aeroacoustics: progress on nonlinear problems of sound generation,” 2004, Progress In Aerospace Sciences.
- ⁴Hesthaven, J. S. and Warburton, T., *Nodal discontinuous Galerkin methods: algorithms, analysis, and applications*, Vol. 54, Springer, New York, 2008.
- ⁵Collatz, L., *The Numerical Treatment of Differential Equations*, Springer-Verlag, New York, 1960.
- ⁶Kreiss, H.-O. and Scherer, G., “Finite element and finite difference methods for hyperbolic partial differential equations,” *Mathematical Aspects of Finite Element in Partial Differential Equations*, Academic Press, Inc., 1974.
- ⁷Mattsson, K. and Nordstrom, J., “Summation by parts operators for finite difference approximations of second derivatives,” *Journal of Computational Physics*, Vol. 199, 2004, pp. 503–540.
- ⁸Colonus, T. and Ran, H., “A super-grid-scale model for simulating compressible flow on unbounded domains,” *J. Comput. Phys.*, Vol. 182, No. 1, 2002, pp. 191–212.
- ⁹Appelö, D. and Colonius, T., “A high-order super-grid-scale absorbing layer and its application to linear hyperbolic systems,” *Journal of Computational Physics*, Vol. 228, No. 11, 2009, pp. 4200–4217.
- ¹⁰Chen, R. and Hagstrom, T., “*P*-adaptive Hermite methods for initial value problems,” *Communications in Computational Physics*, 2010 (submitted).
- ¹¹Team, T. C., “The Chebfun Project,” July 2010.
- ¹²Sommariva, A., Vianello, M., and Zanovello, R., “Adaptive bivariate Chebyshev approximation,” *Numerical Algorithms*, Vol. 38, 2005, pp. 79–94, 10.1007/BF02810617.
- ¹³Berrut, J.-P. and Trefethen, L. N., “Barycentric Lagrange Interpolation,” *SIAM Review*, Vol. 46, No. 3, 2004, pp. pp. 501–517.
- ¹⁴Freund, J., “Noise sources in a low Reynolds number jet at Mach 0.9,” *J. Fluid Mech.*, Vol. 438, 2001, pp. 277–305.